RAPID - A Video Rate Object Tracker

Chris Harris & Carl Stennett

Roke Manor Research Ltd.,
Roke Manor, Romsey, Hampshire, England

RAPID (Real-time Attitude and Position Determination) is a real-time model-based tracking algorithm for a known three-dimensional object executing arbitrary motion, and viewed by a single video-camera. The 3D object model consists of selected control points on high contrast edges, which can be surface markings, folds or profile edges.

The use of either an alpha-beta tracker or a Kalman filter permits large object motion to be tracked and produces more stable tracking results. The RAPID tracker runs at video-rate on a standard minicomputer equipped with an image capture board.

Three-dimensional (3D) Model-Based Vision enables observed image features to be used to determine the pose (ie. position and attitude) of a known 3D object with respect to the camera (or alternatively, the viewpoint of the camera with respect to the model) [1]. The knowledge concerning the object that is used to perform Model-Based Vision is the 3D locations of salient and easily observed object features, such as corners and high-contrast edges (eg. surface marking sand crease edges).

Tracking of a moving object can be achieved by performing such a vision task on successive frames of an image sequence. By making use of the continuity of object motion found in the closely-spaced frames of video imagery, the process of tracking becomes simplified, permitting real-time object tracking to be accomplished. By noting the difference in image position between the observed image features and the projection of the 3D model features with the model in its currently estimated pose, the required small corrections to the object pose can be calculated.

The particular model features that we use to perform tracking are points located on high contrast edges. The projection of these edges into the image is easy to perform, and the corresponding image edges simple to locate by searching the image pixels perpendicularly to the edge direction. Only the perpendicular component of motion is sought because of the aperture problem, and this is why we need the edges to be locally straight. The set of measured displacements of these edges are used to update the estimate of model pose by linearising the resulting equations.

For the edges to be successfully detected, the edges need to be locally straight and uncluttered. Rudimentary edge detection is performed by considering the row of pixels perpendicular to the model edge that pass through the projected control point, and finding the pixel position of the largest brightness gradient along that row.

In general, the projected model control points will not lie precisely on the observed image edges because of errors in the estimate of the object pose due to object motion. However, frames captured at video rate give errors in the object pose which are small so allowing edges to be correctly matched. To first order, small changes in the model pose will cause image displacements of the control points which are linear in the change in the pose, and so the perpendicular distances between the projected control points and the observed image edges will vary linearly with the change in the pose. This linearity enables the small variations of the object pose that serve to minimise these perpendicular distances to be determined by solving a set of simultaneous linear equations, hence producing a fast pose correction. Applying the resulting correction to the pose estimate enables the pose estimate to rapidly converge on the correct object pose as subsequent frames are processed.

If the object is moving across the image, the above method of updating the pose vector will produce a result that lags behind the actual pose. It is advantageous to counter this deficiency by predicting ahead, as inter-frame motion in excess of the half-length of the row of pixels can then be tolerated. This predicting ahead is most simply achieved by using a position and velocity tracker, the so-called alpha-beta tracker [2].

The basic tracker algorithm as described above has limited uses. The necessary enhancements on the basic process described above are:

- to enable and disable control points appropriately as the object attitude varies, and control points are revealed and obscured respectively;
- to ignore control points on weak, error prone edges;
- to define an edge polarity (ie. light/dark or dark/light) so that strong edges with wrong polarity caught in the search region are ignored;
- to make use of profile edges whose 3D control point positions change with object pose (eg. on cylinders, cones, spheres, etc.);
- to predict ahead and to reduce the variability of the pose estimates by using a Kalman Filtering technique[3].

BASIC ALGORITHM

Define the (Cartesian) camera coordinate system, which has its origin at the camera pin-hole, Z-axis aligned along the optical axis of the camera, and X and Y axes aligned along the horizontal (rightward) and vertical (downward) image axes respectively. Imaging of points in 3D will be handled by the introduction of conceptual image-plane situated at unit distance in front of the camera pin-hole. A point at position $\mathbf{R} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ in camera coords will then project to image position $\mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix}$.

Define a model coordinate system, with origin located at **T** in camera coords, and with axes aligned with the camera coordinate system. Consider a control point on the model located at **P** in model coordinates, and situated on a prominent 3D edge. This control point will project onto the

BMVC 1990 doi:10.5244/C.4.15

image at $\mathbf{r} = \frac{1}{(T_z+P_z)} \begin{pmatrix} T_x+P_x \\ T_y+P_y \end{pmatrix}$. Let the tangent to the 3D edge on which the control point is located be called the control edge. The control edge is in practice defined by specifying a companion control point to **P**, often also located on the control edge, and which projects onto the image at **s**. By considering the image displacement between **r** and **s**, the expected orientation of the control edge on the image can be determined. Let this be an angle α from the image x-axis, hence

$$\cos\alpha = \left(\frac{s_x - r_x}{|\mathbf{s} - \mathbf{r}|}\right) \quad , \quad \sin\alpha = \left(\frac{s_y - r_y}{|\mathbf{s} - \mathbf{r}|}\right)$$

We wish to find the perpendicular distance of **r** from the appropriate image edge. Assuming that the orientations of the image edge and the control edge are nearly the same, a one-dimensional search for the image edge can be conducted by looking perpendicularly to the control edge from **r**. However, to search perpendicular for the edge in the image to the edge at the control point would require finding the image intensity at non-pixel positions. To avoid this inconvenience, the image edge is searched for in one of four directions: horizontally, vertically, or diagonally (that is, by simultaneous unit pixel displacements in both the horizontal and vertical directions). If the pixels are square, the diagonal direction will be at 45 degrees, but with different image aspect ratios, other angles will be traversed. The direction which is closest to being perpendicular to the control edge is chosen, and a row of pixel values centred on **r**, the projection of the control point, is read from the image.

Write the orientation of the row of pixels from the x-axis on the image-plane as the angle β, as shown in Figure 1.

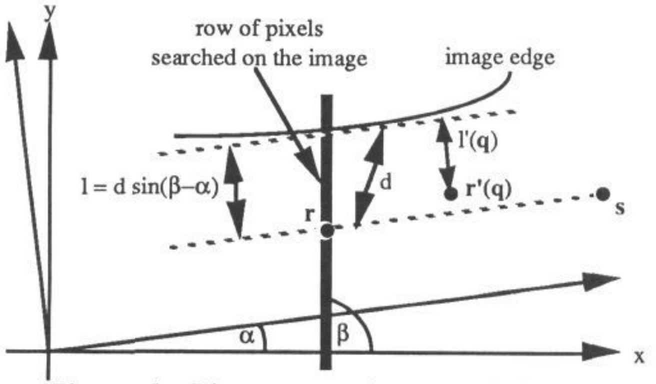

Figure 1: Diagram to show calculation of perpendicular distance, l

On the unit-focal-length image-plane, let the dimensions of a pixel be k_x and k_y in the x and y directions respectively. Hence the orientation of the diagonal directions of the row of pixels will be $\beta = +/- \beta^*$, where $\tan\beta^* = \frac{k_y}{k_x}$. Let the image edge be encountered at a displacement from **r** of n_x pixels in the x-direction and n_y pixels in the y-direction (for diagonal directions, $n_x = \pm n_y$). Then the unit-focal-length image-plane distance of **r** from the image edge along the row of pixels will be

$$d = \sqrt{(n_x^2 k_x^2 + n_y^2 k_y^2)}$$

and the perpendicular distance to the edge will be

$$l = d \sin(\beta-\alpha)$$

Let n be the number of pixel steps (horizontal, vertical or diagonal) traversed along the row of pixels before the edge is encountered. The four permissible orientations of the row of pixels are considered below, together with a condition on the angle α used for selecting a particular orientation:

Horizontal, $\beta = 0$:

$$l = -n\, k_x \sin\alpha \qquad \left|\tan\alpha\right| \geq \tan\left(\frac{\pi}{4} + \frac{\beta^*}{2}\right)$$

Vertical, $\beta = \frac{\pi}{2}$:

$$l = n\, k_y \cos\alpha \qquad \left|\tan\alpha\right| \leq \tan\left(\frac{\beta^*}{2}\right)$$

Upward diagonal, $\beta = \beta^*$:

$$l=n\left(k_y\cos\alpha - k_x\sin\alpha\right) \quad \tan\left(\frac{\beta^*}{2}\right)<\tan\alpha<\tan\left(\frac{\pi}{4} + \frac{\beta^*}{2}\right)$$

Downward diagonal, $\beta =-\beta^*$:

$$l=n\left(k_y\cos\alpha + k_x\sin\alpha\right) \quad -\tan\left(\frac{\beta^*}{2}\right)>\tan\alpha>-\tan\left(\frac{\pi}{4} + \frac{\beta^*}{2}\right)$$

Each control point will result in a measured perpendicular distance, l, as illustrated in Figure 2. The set of these perpendicular distances will be used to find the small change in the object pose that should minimise the perpendicular distances on the next frame processed.

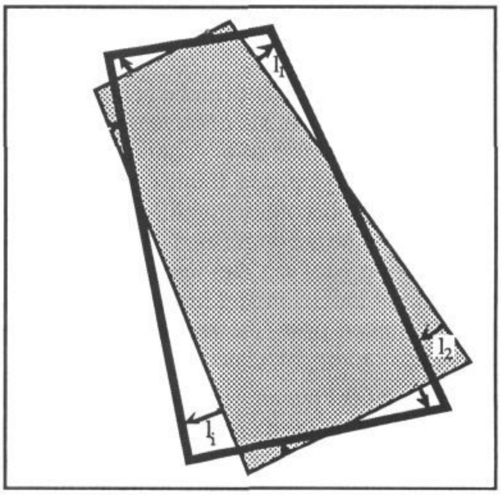

Figure 2: Diagram to show a sample of perpendicular distances, l_i

Consider rotating the model about the model origin by a small angle **Θ**, and translating it by a small distance **Δ**. Write these two small displacements as the six-vector **q**. This will move the model point **P**, located in coords at **R** = **P** + **T**, to **R'** in camera coords.

$$\mathbf{R}'(\mathbf{q}) = \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$
$$\approx \mathbf{T} + \mathbf{\Delta} + \mathbf{P} + \mathbf{\Theta} \times \mathbf{P}$$
$$= \begin{pmatrix} T_x+\Delta_x+P_x+\theta_y P_z-\theta_z P_y \\ T_y+\Delta_y+P_y+\theta_z P_x-\theta_x P_z \\ T_z+\Delta_z+P_z+\theta_x P_y-\theta_y P_x \end{pmatrix}$$

This will project onto the image at

$$\mathbf{r}'(\mathbf{q}) = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \frac{X'}{Z'} \\ \frac{Y'}{Z'} \end{pmatrix}$$

Expanding in small $\boldsymbol{\Delta}$ and $\boldsymbol{\Theta}$, and retaining terms up to first order, gives

$$x' = x + \frac{\left[\Delta_x + \theta_y P_z - \theta_z P_y - x\,(\Delta_z + \theta_x P_y - \theta_y P_x) \right]}{\left[T_z + P_z \right]}$$

$$y' = y + \frac{\left[\Delta_y + \theta_z P_x - \theta_x P_z - y\,(\Delta_z + \theta_x P_y - \theta_y P_x) \right]}{\left[T_z + P_z \right]}$$

where, as before,

$$x = \frac{(T_x+P_x)}{(T_z+P_z)} \quad , \quad y = \frac{(T_y+P_y)}{(T_z+P_z)}$$

Thus $\mathbf{r}'(\mathbf{q})$ can be written

$$\mathbf{r}'(\mathbf{q}) = \mathbf{r} + \begin{pmatrix} \mathbf{q}.\mathbf{a} \\ \mathbf{q}.\mathbf{b} \end{pmatrix}$$

where

$$\mathbf{a} = \frac{1}{T_z+P_z} \begin{pmatrix} -xP_y \\ xP_x + P_z \\ -P_y \\ 1 \\ 0 \\ -x \end{pmatrix} \quad , \quad \mathbf{b} = \frac{1}{T_z+P_z} \begin{pmatrix} -yP_y - P_z \\ yP_x \\ P_x \\ 0 \\ 1 \\ -y \end{pmatrix}$$

Hence the perpendicular distance of the control point from the image edge is

$$\begin{aligned} l'(\mathbf{q}) &= l + \mathbf{q}.\mathbf{a} \sin\alpha - \mathbf{q}.\mathbf{b} \cos\alpha \\ &= l + \mathbf{q}.\mathbf{c} \end{aligned}$$

where

$\mathbf{c} = \mathbf{a} \sin\alpha - \mathbf{b} \cos\alpha$
and l is the measured distance to the edge.

Consider now not just one control point, but N control points, labelled i=1..N. The perpendicular distance of the i'th control point to its image edge is

$l'_i(\mathbf{q}) = l_i + \mathbf{q}.\mathbf{c}_i$

We would like to find the small change of pose, $\mathbf{q}$, that aligns the model edges precisely with the observed image edges, that is to make all $l'_i(\mathbf{q})$ zero. If the number of control points, N, is greater than 6, then this is not in general mathematically possible as the system is overdetermined. Instead, we choose to minimise, E, the sum of squares of the perpendicular distances

$$E(\mathbf{q}) = \sum_{i=1}^{N} \left[l_i + \mathbf{q}.\mathbf{c}_i \right]^2 .$$

By setting to zero the differentials of E with respect to $\mathbf{q}$, the following equations are obtained

$$\sum_{i=1}^{N} \left(\mathbf{c}_i\, \mathbf{c}_i^T \right) \mathbf{q} = - \sum_{i=1}^{N} \left(l_i\, \mathbf{c}_i \right)$$

This is a set of 6 simultaneous linear equations, and so can be solved using standard linear algebra.

The change, $\mathbf{q} = \begin{bmatrix} \boldsymbol{\Theta} \\ \boldsymbol{\Delta} \end{bmatrix}$, in the model pose specified by the above algorithm must now be applied to the model. Applying the change in model position is straightforward

$\mathbf{T} := \mathbf{T} + \boldsymbol{\Delta}$

The change in object attitude however causes some difficulties. Conceptually, the positions of the control points on the model should be updated thus

$\mathbf{P}_i := \mathbf{P}_i + \boldsymbol{\Theta} \times \mathbf{P}_i$

However, after numerous (thousands of) cycles of the algorithm, finite numerical precision and the approximation to rotation represented by the above equation, results in the control points no longer being correctly positioned with respect to each other, and thus the model distorts. To overcome this problem, the attitude of the model is represented by the rotation vector $\boldsymbol{\phi}$ (a 3-vector whose direction is the axis of rotation and whose magnitude is the angle of rotation about this axis), which rotates the model from its reference attitude, in which the model has its axes aligned with the camera coordinate axes. From the rotation vector $\boldsymbol{\phi}$ can be constructed the orthonormal rotation matrix $A(\boldsymbol{\phi})$, which appropriately rotates any vector to which it is applied. Conceptually, the rotation vector, $\boldsymbol{\phi}$, should be updated by the model attitude change, $\boldsymbol{\Theta}$, thus

$A(\boldsymbol{\phi}) := A(\boldsymbol{\Theta})\, A(\boldsymbol{\phi})$

but doing this, the orthonormality of the rotation matrix may be lost in time, so in practice the rotation vector, $\boldsymbol{\phi}$, is updated directly by use of quaternions. If $A(\boldsymbol{\phi})$ is the rotation matrix after updating, and the i'th model point is located in reference coordinates at $\mathbf{P}_i^{(ref)}$, then the position of this point in model coordinates at the beginning of the next cycle will be

$\mathbf{P}_i = A(\boldsymbol{\phi})\, \mathbf{P}_i^{(ref)}$.

ALPHA-BETA TRACKER

When using only the previous pose estimate in RAPID, the tracking lagged behind the actual position of the model, always having to catch up. A tracker was needed to predict ahead the motion of the model to give RAPID the ability to cope with fast but smooth motion across the image. The alpha-beta tracker is one such tracker and was one of the simplest to implement.

Let $\mathbf{x}_t$ be the 6-vector that represents the estimate of the pose of the object at frame number t, and $\mathbf{q}_t$ the change in pose requested by the above algorithm. The alpha-beta tracker is given by

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{v}_t + \alpha\, \mathbf{q}_t \\ \mathbf{v}_{t+1} &= \mathbf{v}_t + \beta\, \mathbf{q}_t \end{aligned}$$

where the 6-vector $\mathbf{v}_t$ is the estimated velocity, that is, the rate of change of pose per frame processed. α and β are scalars effecting the integration time of the predictor.

On each image processed, the change of pose, $(\boldsymbol{\Theta}, \boldsymbol{\Delta})$, desired by the algorithm has a very prompt responsiveness, and this leads to noisy pose estimates. Use of the alpha-beta tracker can serve to smooth out this response, but in some circumstances such an ad hoc approach results in instability. These can be overcome by properly modelling the kinematics of the object by using a Kalman Filter, as described by Evans[4].

ENHANCEMENTS TO THE BASIC ALGORITHM

Enabling and Disabling Control Points

When the tracked object moves, control points are often self-obscured, for example by rotating around to the back side of the object. Such control points need to be disabled as they become invisible, or else they will pick up on incorrect edges, and interfere with the tracking. To cope with this problem, a quantised view-potential table is constructed, corresponding to the different distant camera viewpoints with which the camera can view the object, and the visibility of each control point is stored as a binary flag in the table. The view-potential table is based upon square bins on the faces of a cuboid box surrounding the object, each face being subdivided into 144 bins. Assignment of the true/false visibility flags is currently user-controlled as the object is moved, though if a CAD model of the object were available, ray-caching could construct them automatically.

Ignoring Control Points On Weak Edges

If the edge response at a control point becomes too weak, it is dangerous to use it for tracking, as it may subsequently incorrectly latch on to a stronger nearby edge. For this reason the control point is automatically disabled when the edge response is small, and re-enabled when it becomes large. A common reason for weak edge responses is obscuration by natural passing before the camera, such as somebody's moving hand.

Defining the Edge Polarity

The polarity of an edge is whether it is a dark/light edge or a light/dark edge. The polarity is a very useful device to reject incorrect edges, as nearby edges on an image very often have opposite polarities. The expected edge polarity at each camera viewpoint could easily be included in the aforementioned visibility table.

Tracking Conic Profile Edges

The theory presented above is only able to track control points situated on edges which have an objective 3D existence, such as surface markings and folds. We wish to extend the theory to cater for profile edges, as they are often very prominent on images, and objective 3D edges may be deficient. The simplest case to consider is that of a surface of revolution, and fortunately, these are common in man-made objects.

Consider, for example, making use of the two profile edge points **P** and **P'** lying on the surface S shown in Figure 3. These points lie on the circle of intersection of the plane passing through the point **O**, that is perpendicular to the axis of revolution, **AA'**. Now the points **P** and **P'** will also lie on the cone C that is tangential to the circle of intersection, and co-axial with the surface, S. This cone is specified by its apex, **A**, the centre of the circle of intersection, **O**, and the radius of the circle, r. From this specification, the 3D locations of the profile points **P** and **P'** can readily be obtained. Consider now the 3D lines **AP** and **AP'** which lie on the surface of the cone. To first order, changes in pose of the cone will leave these lines lying on the profile edge. Thus by simply using **P** and **P'** as the control points with companion point **A**, the profile edge at **P** and **P'** can be introduced into the formulation constructed for objective edges.

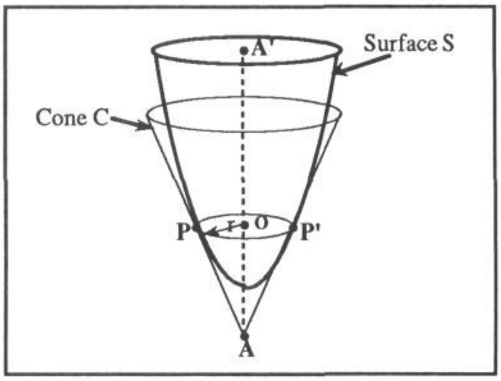

Figure 3: Diagram to illustrate theory of profile edges

The profile edge theory has been further generalised to cater for general quadric surfaces (eg. spheres, ellipsoids and hyperboloids) so permitting a large class of profile edges to be tracked using RAPID.

LABORATORY DEMONSTRATION

For a demonstration of the algorithm, the following figures show RAPID tracking a number of objects using an image capture board driven by a microVAX 3400.

Figure 4 shows the RAPID tracking of the box in a static situation, with the box outline being displayed. The control points used in the tracking are shown by the white dots situated on the outline and elsewhere. Displaying a large outline like this slows RAPID down by about a factor of two.

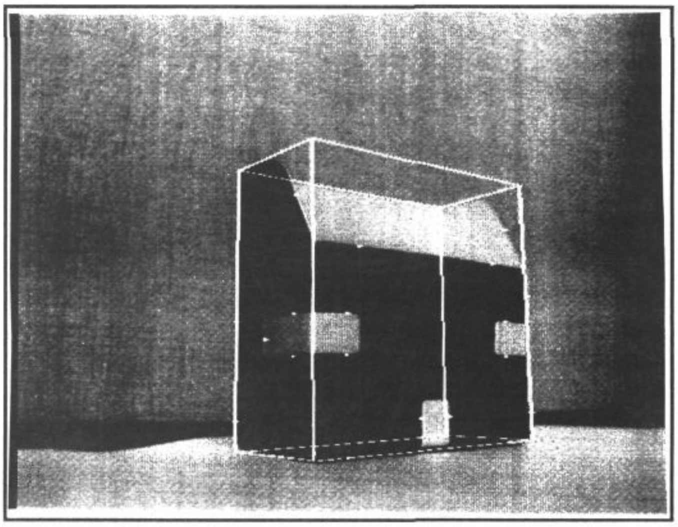

Figure 4: RAPID tracking box in a static situation

Figure 5 shows the box being moved by hand. Note that the point obscured by the hand is ignored, as the algorithm does not find the strong edge it is looking for. This makes the algorithm robust in this respect as it will work as long as there are enough points with strong edges to give a well-conditioned pose estimate.

Figure 6 shows the tracking of a cone using conical profile edges. In situations like this, RAPID can track more robustly given points on profile edges.

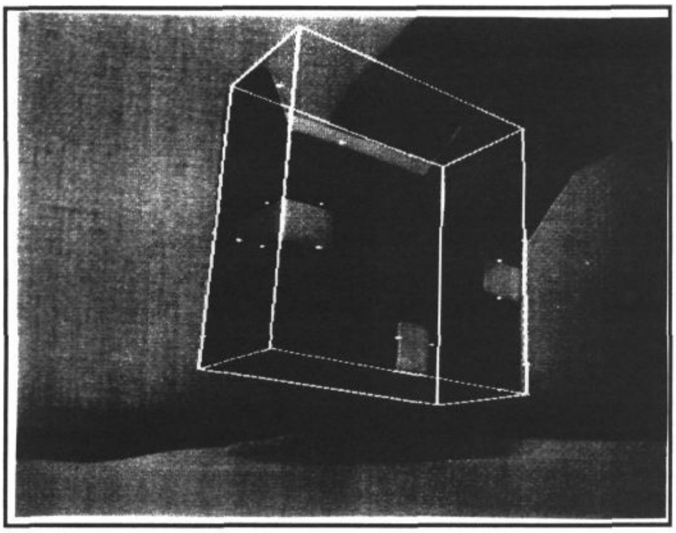

Figure 5: RAPID tracking a moving box

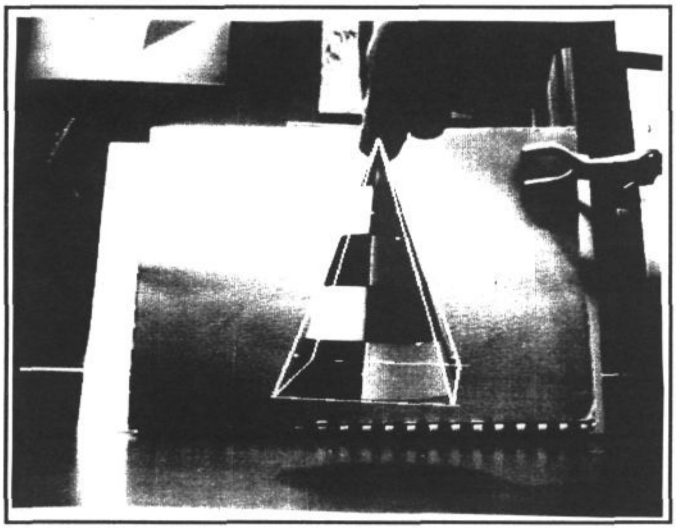

Figure 6: RAPID tracking a cone making use of conical profile edges

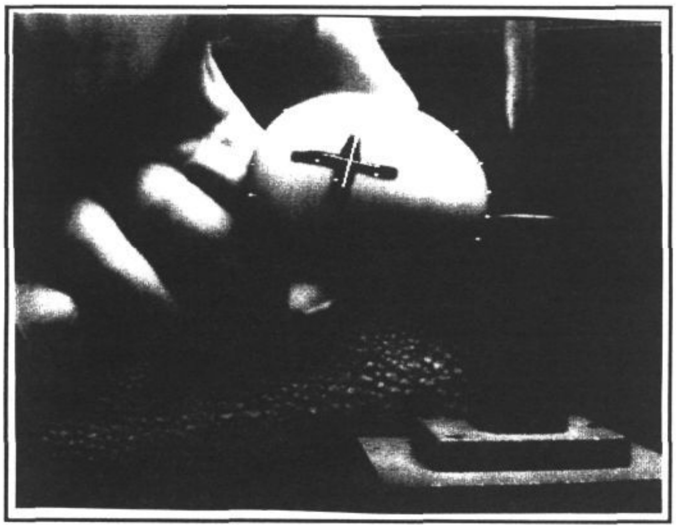

Figure 7: RAPID tracking an egg making use of ellipsoidal profile edges

Figure 7 shows RAPID tracking an egg using profile edges on ellipsoids. As the egg model is a surface of revolution, a black cross marked on the egg is needed to give RAPID the missing information it needs about rotation of the egg about its axis of revolution. Without at least one control point on the horizontal bar of the cross, the egg model starts spinning about the axis of revolution in a random manner. However, given the cross on the egg, RAPID tracks the egg without problems.

CONCLUSION

Tracking of a 3D object using selected control points on both objective and profile edges has proved to be both robust and fast. Linearising the small changes in model pose has proved to be practicable, and a single cycle of convergence on each frame sufficient.

Similar work has been pursued by Stephens[5] and Bray[6], but RAPID genuinely runs at video rate and produces accurate pose estimates on very modest hardware (currently a multiuser microVax 3400 with an image capture board). Execution time is split up into two main areas: computational time; and image capture board processing time. These two temporal areas are roughly equal with the Kalman filter on. Improvements in interfacing with the image capture board would significantly improve the performance of RAPID.

The main difference between RAPID and similar work of Bray[6] is in the edge detection method, the single iteration of the convergence procedure, and the extension to profile edges.

Rather than performing a least squares fit to a whole edge line, RAPID uses only a few selected control points along the edge. At these control points, the search region for the edge is approximately perpendicular to the edge. (This edge detection method is similar to Stephens' edge detection method except that convolutions are not used due to the expense of processor time.) Although using the whole edge might make the algorithm more robust in some instances, good placement of control points maintains the robustness of the algorithm and gives RAPID its speed.

A single iteration of the convergence procedure has proven to be sufficient because if there is a large change of pose such that a single iteration would be insufficient, then the edges that RAPID is trying to find on the image will either have disappeared or be out of reach of the pixel detection row. Hence tracking will cease or will be incorrect.

REFERENCES

1 **D G Lowe** "Stabilized Solution for 3D Model Parameters" First European Conference on Computer Vision, ECCV90.

2 **S S Balckman** "Multiple-Target Tracking with Radar Applications" Artech House Inc, 1986.

3 **R E Kalman** "A new approach to linear filtering and prediction problems" Trans. ASME, J. of Basic Engineering, March 1960.

4 **R J Evans** "Filtering of Pose Estimates Generated by the RAPID Tracker in Applications" Proceedings of the first British Machine Vision Conference BMVC90.

5 **R S Stephens** "Real-Time 3D Object Tracking" Proceedings of the fifth Alvey Vision Conference, AVC89.

6 **A J Bray** "Tracking objects using image disparities" Proceedings of the fifth Alvey Vision Conference, AVC89.